제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003.5)

표준 해쉬 알고리즘 HAS-160의 설계

임재청, 송문빈, 박상원, 심정섭, 정연모
경희대학교 전자공학과
e-mail : chung@khu.ac.kr

Design of Standard Hash Algorithm HAS-160

Jaechung Lim, Moonvin Song, Sangwon Park, Jungsub Sim, Yunmo Chung
Dept. of Electronic Engineering, Kyung-Hee University

요 약

본 논문은 인터넷 보안 및 인증에 널리 사용되는 해쉬 알고리즘인 HAS-160을 하드웨어로 구현 하였다. 구현을 위해서는 VHDL을 사용하여 모델링 하였으며 또한 단계연산에 필요한 연산들의 최적화된 스케줄링으로 동작속도를 증가 시켰다.

1. 서론

혁신적인 통신기술의 발전은 우리생활에 많은 변화를 가져왔다. 인터넷 뱅킹, 전자상거래, 사이버 주식거래 등이 보편화되었으며, 따라서 중요한 개인정보와 온라인 뱅킹 등을 통한 자금의 이동이 이루어지고 있다. 또한 PDA, 셀룰러폰, IMT2000등 개인 휴대용 모바일 등에서도 확산될 것이다.

이러한 서비스를 구현하기 위해서는 전송 매체의 발전과 더불어 개인정보의 암호화 기술 및 전자서명 기술들이 필요하다. 그러나 컴퓨터의 발달과 암호화 기술에 상응하는 공격기술의 발달로 더 높은 수준의 암호화 기술을 끊임없이 요구한다. 하지만 이러한 기술의 복잡도가 높아짐에 따라 구현되는 장비에 처리속도 및 비용 등의 많은 부담을 가져오게 된다. 이를 해결하기 위해서 암호화 알고리즘을 하드웨어화 한다.

해쉬 알고리즘은 인터넷상에서 데이터의 무결성, 인증, 부인방지, 재사용 방지 등의 목적에 사용된다. 이는 인터넷 전자금융, 전자상거래, P2P (Peer to Peer)서비스에 필수 요건이다. 각종 통신 장비나 보안관련 플랫폼에 소프트웨어로 구현된 예는 있지만 하드웨어 화된 사례는 없다. 본 논문에서는 대한민국 표준 해쉬 알고리즘인 HAS-160을 VHDL로 모델링 한 후 FPGA로 구현하여 기능을 검증하였다.

2. HAS-160 알고리즘

HAS-160 알고리즘은 임의의 길이의 입력 데이터를 512비트의 크기를 가지는 블록 단위로 처리하여 160비트의 최종 출력데이터를 생성하는 알고리즘이다 [1]. 해쉬 알고리즘은 다음과 같은 조건을 만족해야 한다.

1) 주어진 출력에 대하여 입력 값을 구하는 것이 계산상 불가능하다.
2) 주어진 입력에 대하여 같은 출력을 내는 또 다른 입력을 찾아내는 것이 계산상 불가능하다.
3) 같은 출력을 내는 임의의 서로 다른 두 입력 메시지를 찾는 것이 계산상 불가능하다.

전체 입력을 512비트의 블록 단위로 입력하고, 마지막 데이터 블록의 길이는 448비트가 되도록 '0'으로 채운다. 마지막 64비트는 '0'을 채우기 전에 메시

지 길이를 계산하여 채워 넣는다. [그림1]은 HAS-160의 구조를 나타낸 그림이다.

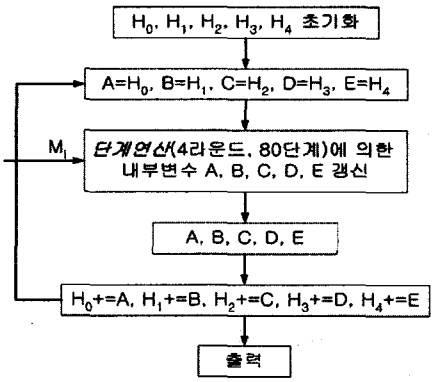

그림 1. HAS-160의 구조

HAS-160에서 사용되는 연쇄변수 H_n은 H_0 = 67452301, H_1 = efcdab89, H_2 = 98badcfe, H_3 = 10325476 및 H_4 = c3d2e1f0으로 각각 초기화된다. 단계연산은 총 4라운드 80단계의 연산을 거치며 이 과정에서 메시지 변수 M_i를 받아 메시지들을 처리한다. 메시지 변수 M_i는 입력받은 512비트의 메시지를 비트열-워드열 변환규칙(little-endian convention)을 따라 16개의 32비트 워드로 변환한다. 16개의 워드로부터 4개의 메시지 변수 X[16], X[17], X[18], X[19]를 생성한다. 생성되는 워드는 각 라운드의 0-4, 6-9, 11-14 및 16-19 단계 연산들에서 사용되는 메시지 변수들의 XOR 값으로 계산된다. 각 단계연산에서 사용되는 메시지 변수의 값은 [표1]과 같다.

표 1. 단계연산에 사용되는 메시지 변수

i	라운드 1	라운드 2	라운드 3	라운드 4
0	X[18]	X[18]	X[18]	X[18]
1	X[0]	X[3]	X[12]	X[7]
2	X[1]	X[6]	X[5]	X[2]
3	X[2]	X[9]	X[14]	X[13]
4	X[3]	X[12]	X[7]	X[8]
5	X[19]	X[19]	X[19]	X[19]
6	X[4]	X[15]	X[0]	X[3]
7	X[5]	X[2]	X[9]	X[14]
8	X[6]	X[5]	X[2]	X[9]
9	X[7]	X[8]	X[11]	X[4]
10	X[16]	X[16]	X[16]	X[16]
11	X[8]	X[11]	X[4]	X[15]
12	X[9]	X[14]	X[13]	X[10]
13	X[10]	X[1]	X[6]	X[5]
14	X[11]	X[4]	X[15]	X[0]
15	X[17]	X[17]	X[17]	X[17]
16	X[12]	X[7]	X[8]	X[11]
17	X[13]	X[10]	X[1]	X[6]
18	X[14]	X[13]	X[10]	X[1]
19	X[15]	X[0]	X[3]	X[12]

[그림2]는 단계연산의 구조를 나타낸 그림이다.

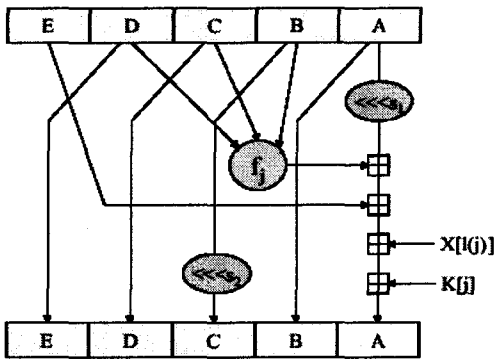

그림 2. 단계연산

[그림2]에서 S_1과 S_2는 j(1≤j≤80)번째 단계연산에 사용되는 순환이동이며, 순환이동의 양은 4라운드 80단계마다 재 정의된다. S_1의 순환이동은 $S_1(j)$ = 5, 11, 7, 15, 6, 13, 8, 14, 7, 12, 9, 11, 8, 15, 6, 12, 9, 14, 5, 13 (0≤j≤19, 20≤j≤39, 40≤j≤59, 60≤j≤79) 이고 S_2는 $S_2(j)$=10 (0≤j≤19), $S_2(j)$=17 (20≤j≤39), $S_2(j)$=25 (40≤j≤59) 및 $S_2(j)$=30 (60≤j≤79)이다. [그림 2]의 f_j는 각 단계연산에서 사용하는 부울 함수이다.

하나의 512비트 데이터 블록에 대해 4라운드, 80단계의 연산이 끝나면 그 결과 A, B, C, D, E를 연쇄변수 H_0~H_4에 더하여 H_0+=A, H_1+=B, H_2+=C, H_3+=D, H_4+=E와 같이 갱신한다.

3. 구현

[그림 3]은 HAS-160의 전체 블록도 이다.

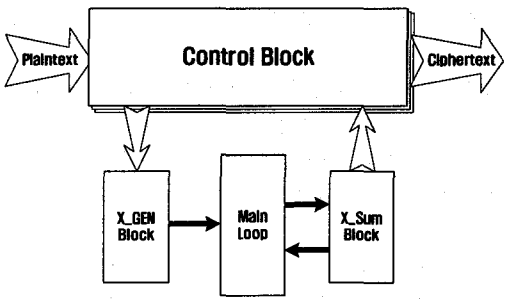

그림 3. 시스템 블록 다이어그램

시스템은 4개의 기능블록으로 구성되어 있으며 데

이터를 입력받는 Control 블록은 현재 수행되는 상태를 제어하고, 외부로부터 입력받는 데이터를 X_Gen블록으로 넘겨준다. 계산된 해쉬 값을 외부로 출력하며 전체 시스템의 동작과 입력 및 출력을 컨트롤한다.

X_Gen블록에서는 Control 블록에서 입력받은 데이터로부터 새로운 데이터 블록을 4라운드 80단계에 따라 생성하여 Main_Loop 블록으로 전송한다.
[그림 4]는 X_Gen 블록을 나타낸 것이다.

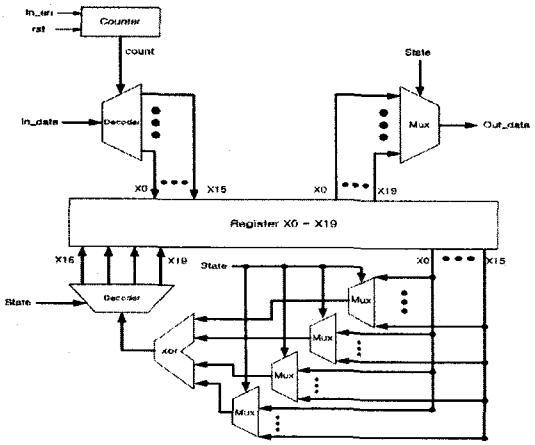

그림 4. X_Gen Block의 세부구조

X_Gen블록 내부에는 외부로부터 입력 받은 데이터를 저장하는 16개의 레지스터가 있으며, 이로부터 각 라운드마다 생성하는 데이터를 저장하는 4개의 레지스터가 있다.
[그림 5]는 Main_Loop 블록을 나타낸 것이다.

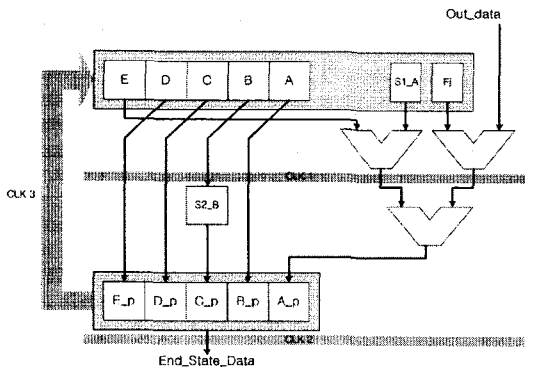

그림 5. Main_Loop Block의 세부구조

Main_Loop 블록에서 단계의 연산을 수행하기 위해서는 3클럭이 필요하다. 최초 [그림 3]의 X_Sum 블록으로부터 초기화 상수를 받아 A, B, C, D, E를 초기화 시킨 후 단계연산이 시작되며, 각 단계에 필요한 메시지 변수는 X_Gen 블록에서 입력받는다. 단계연산을 처리하는 Main_Loop 블록에서는 4번의 32비트 덧셈과정과 1번의 논리연산 및 2번의 비트순환 연산을 수행한다. 이때 최장지연경로가 발생한다. 여기서 발생한 최장지연경로를 최소화하기위해 각각의 단계연산에 소요되는 클럭을 분배하여 스케줄링한다.

512비트의 메시지가 80단계를 거쳐 X_Sum 블록에서 End_State_Data 값을 출력한다. 입력 데이터가 다수의 512비트로 구성되어 있으면, 초기화 상수를 X_Sum 블록에서 받아 다시 80단계를 처리한다. 이때의 초기화 상수는 이전 512비트로부터 누적 연산된 결과 값이다.

X_Sum 블록은 초기화 상수 Reset_Data를 생성하여 Main_Loop 블록으로 Setup_Data를 전송한다. 이때 초기화 되는 값은 HAS-160 해쉬 알고리즘에서 H0 = 67452301, H1 = efcdab89, H2 = 98badcfe, H3 = 10325476 및 H4 = c3d2e1f0로 정의된 16진 상수이다.

[그림6]은 X_Sum 블록의 세부구조를 나타낸 것이다.

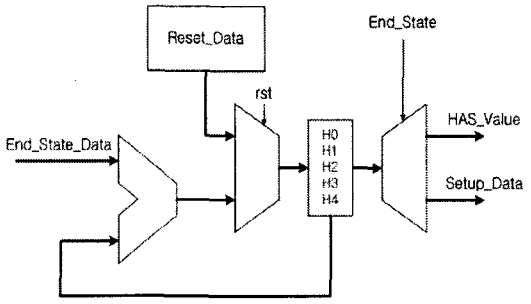

그림 6. X_Sum Block의 세부구조

80단계의 연산을 거친 Main_Loop 블록의 결과값 End_State_Data를 누적 연산하면, 160비트가 생성된다. 생성된 해쉬값 HAS_Value를 Control 블록으로 전송한다. 추가적인 512비트의 데이터블록이 있다면, Main_Loop블록을 다시 초기화시키기 위한 값은 이전 80단계의 결과 값을 이용하여 초기화 한다.
더 이상의 메시지가 없다면 Control Block으로 부터 End_State 신호가 발생하여 최종 해쉬 값인

HAS_Value 값이 Control Block으로 전송된다. 전송된 해쉬 값은 컨트롤 블록에 의하여 32비트 버스를 통해 출력한다.

[그림 7]은 전체 블록의 합성도이다.

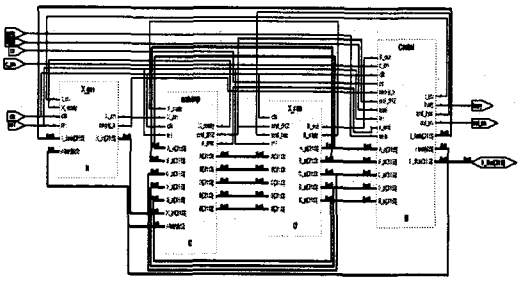

그림 7. Top Block의 합성도

합성 과정은 Altera의 APEX-II 디바이스에서 하였으며 구현 결과 동작 속도는 110MHz이다. 세부적인 합성결과는 [표2]와 같다.

표 2. 세부합성결과

Total LUTs	2494 of 16640 (14%)
Logic resources	3007 ATOMs of 16640 (18%)

한 단계연산에 3클럭이 소요되며, 512비트를 처리하기위해서는 80단계의 수행과정을 거치므로 총 240 클럭이 단계연산에 필요하다.

입력 메시지가 512비트의 경우 단계연산을 수행하기 위한 클럭과 최종 덧셈을 위한 클럭, 메시지변수 X[16], X[17], X[18], X[19]를 생성하기위한 클럭의 소요시간은 모두 2.345μs이다. 이는 218.3Mbps의 처리속도이다.

[표3]은 MD5 해쉬 알고리즘의 구현 시스템에 따른 속도를 나타내는 표이다.

표 3. MD5 구현 시스템에 따른 성능

MD5 수행능력		속도
Software	Intel Pentium/90 NeXTStep[3]	44Mbps
	Sun SPARC-20/71 SunOS 4.1.3[3]	57Mbps
Hardware	Xilinx Virtex[4]	165Mbps
	ALTERA APEX20k[5]	208Mbps

HAS-160은 MD5의 4라운드 64단계와 비교하여 더욱 복잡한 구조와 많은 단계를 수행한다. 그러나 하드웨어로 구현한 결과 218.3Mbps로 더 좋은 성능을 갖는다.

4. 결 론

본 논문에서는 국내 표준 해쉬 알고리즘인 HAS-160을 VHDL로 모델링 하여 FPGA로 검증하였다. 단계연산에 필요한 덧셈, 논리연산, 비트순환 과정을 개별 적으로 모델링한 후 합성하는 방법을 사용하였다. 단계연산은 내부적으로 3단계의 처리 과정을 거쳐 연산을 수행하는 구조를 갖는다.

참고문헌

[1] TTAS.KO-12.0011/R1(2000) : Hash function Algorithm Standard (HAS-160)

[2] Douglas R. Stinson, *Cryptography Theory and Practice*, CRC, 1995.

[3] J. Touch, *"Report on MD5 performance"*, RFC 1810, June, 1995.

[4] Janaka Deepakumara, Howard M. Heys and R. Venkatesan, "FPGA IMPLEMENTATION OF MD5 HASH ALGORITHM" *Technical Paper* , Engineering and Applied Science Memorial University of Newfoundland St.John Canada, 2001.

[5] 윤희진, 정용진 "CSA를 사용한 고속 MD5프로세서 구현", 한국정보처리학회, 2002.

[6] Warwick Ford, *Computer Communications Security*, Prentice-Hall International Inc, 1994.

[7] Man Young Rhee, *Cryptography and Secure Communications*, McGraw-Hill, 1994.

[8] Bruce Schneier, Applied *Cryptography*, John Wiley & Sons Inc, 1996.